CoEncode: Reclaiming Idle Encoder GPUs in Multi-Tenant Multimodal LLM Serving

Mona Moghadampanah*, Xiangchen Li*, Cheol-Ho Hong†, and Dimitrios S. Nikolopoulos*
* Department of Computer Science, Virginia Tech, Blacksburg, VA, USA
{monamp, lixiangchen, dsn}@vt.edu
† Chung-Ang University, Seoul, Republic of Korea
cheolhohong@cau.ac.kr

***Abstract*—Disaggregated multimodal LLM serving improves stage isolation but can leave dedicated encoder GPUs substantially underutilized. We present CoEncode, a workload-aware encoder pooling approach that uses model-specific capacity profiles to assign fractional streaming multiprocessor (SM) resources and colocate encoder workloads on a shared GPU. CoEncode accounts for both encoder demand and end-to-end pipeline performance when determining resource allocations. Our evaluation shows that CoEncode retains 99.4% of dedicated SLO goodput while improving SLO goodput per GPU by 19.3% and reducing encoding energy consumption by 45.9%. These results demonstrate the potential of encoder pooling to improve resource and energy efficiency while maintaining SLO-compliant end-to-end performance.**

***Index Terms*—multimodal LLM serving, disaggregated serving, GPU sharing, encoder colocation, resource management**

I. Motivation

Stage-disaggregated multimodal LLM (MLLM) serving separates vision encoding (E), prefill (P), and decoding (D) across GPUs to isolate their resource requirements [2]. However, dedicated encoder GPUs are often substantially underutilized, making E+P+D serving resource-inefficient under time-varying multi-tenant demand.

One way to reduce GPU count is to colocate the encoder with prefill or decode [1]. Our measurements show that such cross-stage colocation incurs substantial latency penalties. At 4 requests/s, encoder–prefill colocation nearly doubles p95 TTFT, while encoder–decode colocation increases p95 TPOT by about 85%.

CoEncode instead explores intra-stage colocation, where encoders from different tenants share a GPU while prefill and decode remain isolated. This reclaims underutilized encoder capacity without exposing latency-critical language-model stages to cross-stage interference.

II. Workload and Encoder Characterization

We first examine a representative stage-disaggregated E+P+D run to compare encoder resource use with the language-model stages. As shown in Fig. 1, the encoder averages only 10.8% steady-state streaming multiprocessor (SM) activity, versus 60.7% for prefill and 63.3% for decode, revealing substantial underutilization of the dedicated encoder GPU.

To determine whether this capacity can be safely reclaimed, we characterize encoder capacity under partial GPU allocation

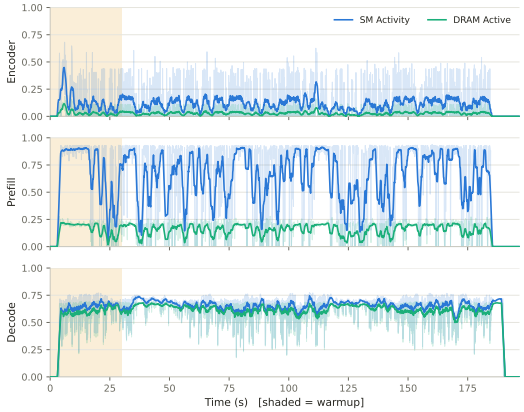


Fig. 1: Stage-level GPU activity in E+P+D serving for InternVL2.5-8B at 4 requests/s.

and test whether the resulting profiles remain valid under colocation. We define sustainable capacity as the maximum delivered visual-token throughput at saturation. Fig. 2 shows that capacity increases with SM share and varies across models, motivating model-specific provisioning. Across tested shares, colocated capacity remains within 10% of the corresponding solo profile, supporting its use for CoEncode's allocation decisions.

III. Experimental Setup

We evaluate CoEncode on an NVIDIA A100 80 GB using Qwen2.5-VL-7B and InternVL2.5-8B. ServeGen-derived [3] open-loop workloads use E+P+D. Encoder shares are enforced with Multi-Process Service (MPS) and disjoint texture processing cluster (TPC) masks via libsmctrl; unlike Multi-Instance GPU (MIG)'s coarse partitions, this supports asymmetric allocations. Capacity is measured at saturation; end-to-end runs use 4 requests/s after warm-up. TTFT SLOs are set to 1.1× the dedicated E+P+D p95 TTFT, yielding 2.3 s for Qwen and 3.2 s for InternVL. Energy is measured with Zeus/NVML. All configurations use three trials; error bars show the min–max range.

IV. CoEncode

We propose CoEncode, a workload-aware encoder pooling system for multi-tenant disaggregated MLLM serving. CoEn-

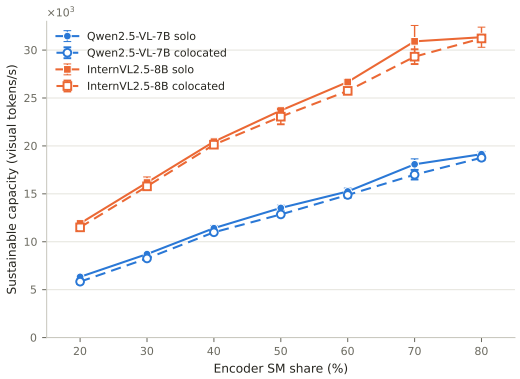


Fig. 2: Solo and colocated encoder capacity across SM shares. Colocated capacity closely tracks the solo capacity profile for both models, remaining within 10% across the tested allocations.

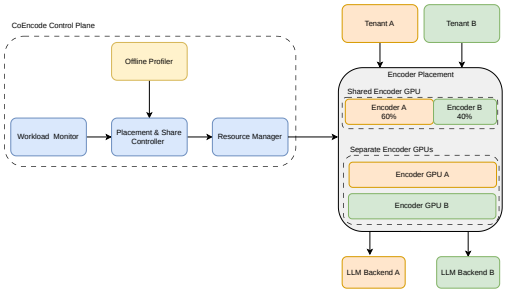


Fig. 3: CoEncode uses workload demand and offline capacity profiles to select encoder placement and SM shares while maintaining end-to-end SLOs.

code allocates SM resources based on each model's demand and capacity profile and colocates encoders when they can safely share a GPU, improving resource and energy efficiency while meeting throughput and end-to-end TTFT targets.

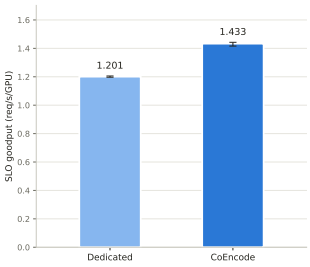
CoEncode monitors each tenant's visual-token arrival rate. For encoder i, an offline capacity profile $C_i(s)$, derived from solo measurements, estimates the maximum sustainable visual-token throughput at SM share s. Given demand λ_i, CoEncode selects the smallest share that can support it:

$$s_i^{\min} = \min\{s \mid C_i(s) \geq \lambda_i\}. \quad (1)$$

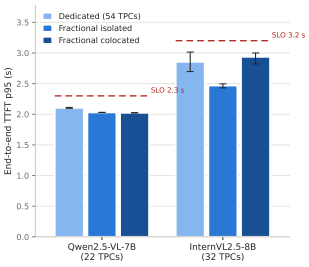
Capacity alone is insufficient for allocation decisions. CoEncode also considers downstream backpressure and end-to-end TTFT when selecting SM shares and determining whether encoders can safely share a GPU. If no shared configuration satisfies the performance targets, the encoders are placed on separate GPU resources.

V. Results

Our results show that CoEncode can consolidate heterogeneous encoder workloads while maintaining end-to-end performance targets. As shown in Fig. 4b, p95 TTFT remains



(a) SLO goodput per GPU for dedicated and colocated encoders.



(b) p95 TTFT across dedicated, fractional-isolated, and fractional-colocated configurations.

Fig. 4: Performance comparison across dedicated and fractional encoder configurations.

within the per-model SLOs under the evaluated colocated configurations, despite modest latency increases relative to dedicated serving.

CoEncode retains 99.4% of dedicated SLO goodput while improving SLO goodput per GPU from 1.201 to 1.433 req/s/GPU, a 19.3% improvement (Fig. 4a). Encoder energy consumption decreases from 83.44 to 45.11 J/1K visual tokens under colocation, a 45.9% reduction relative to dedicated provisioning. These results show that encoder pooling can improve resource and energy efficiency while maintaining SLO-compliant end-to-end performance.

VI. Conclusion and Ongoing Work

Our results show that CoEncode reclaims underutilized encoder capacity using model-specific profiles while preserving end-to-end SLOs. Ongoing work focuses on scaling encoder pooling to larger tenant sets, interference-aware placement, SLO-constrained SM-share and energy optimization, and runtime adaptation to workload demand and pipeline backpressure.

Acknowledgements

This work was supported by NSF Grant No. 2315851, a Sony Faculty Innovation Award, an Amazon Research Award, and the Virginia Tech College of Engineering. OpenAI ChatGPT was used for language editing; the authors are responsible for all technical content, analysis, and results.

References

[1] Xianzhe Dong et al. "HydraInfer: Hybrid disaggregated scheduling for multimodal large language model serving". In: *arXiv preprint arXiv:2505.12658* (2025).

[2] Haoran Qiu et al. "Modserve: Scalable and resource-efficient large multimodal model serving". In: *arXiv preprint arXiv:2502.00937* (2025).

[3] Yuxing Xiang et al. "ServeGen: Workload Characterization and Generation of Large Language Model Serving in Production". In: *arXiv preprint arXiv:2505.09999* (2025).